Ministry of Higher Education and Scientific Research - Iraq
University of Diyala
College of Engineering for Artificial Intelligence Technology
Department of Cybersecurity Engineering

الملحق ٤: وصف المادة الدراسية

MODULE DESCRIPTION FORM

نموذج وصف المادة الدراسية

Module Information معلومات المادة الدراسية					
Module Title	**Data Structures and Algorithms**			**Module Delivery**	
Module Type	**Core**			☒**Theory** ☒**Lecture** ☒ **Lab** ☒**Tutorial** ☐**Practical** ☐**Seminar**	
Module Code	**CSE 208**				
ECTS Credits	**5**				
SWL (hr/sem)	**125**				
Module Level	2		**Semester of Delivery**	4	
Administering Department	Cybersecurity Eng.		**College**	College of Artificial Intelligence Engineering Technology	
Module Leader	Farah Hatem Khorsheed		**e-mail**	farah_hatam@uodiyala.edu.iq	
Module Leader's Acad. Title	Lecturer		**Module Leader's Qualification**	MSc	
Module Tutor	Name (if available)		**e-mail**	E-mail	
Peer Reviewer Name			**e-mail**		
Scientific Committee Approval Date	14/05/2026		**Version Number**	1.0	

Relation with other Modules العلاقة مع المواد الدراسية الأخرى			
Prerequisite module	CSE 102	**Semester**	2
Co-requisites module	None	**Semester**	

Ministry of Higher Education and Scientific Research - Iraq
University of Diyala
College of Engineering for Artificial Intelligence Technology
Department of Cybersecurity Engineering

Module Aims, Learning Outcomes and Indicative Contents أهداف المادة الدراسية ونتائج التعلم والمحتويات الإرشادية	
Module Objectives أهداف المادة الدراسية	By the end of this module, students will be able to: 1. Understand the fundamental concepts of data structures and algorithms. 2. Explain the purpose and characteristics of common data structures. 3. Develop basic Python programs for representing and manipulating data structures. 4. Understand the concepts and operations of arrays, stacks, queues, linked lists, trees, and graphs. 5. Implement fundamental data structures using Python. 6. Understand the basic principles of sorting algorithms. 7. Implement Bubble Sort and Merge Sort using Python. 8. Analyze basic algorithmic approaches and compare their suitability for different problems. 9. Develop problem-solving skills through the practical application of data structures and algorithms. 10. Apply appropriate data structures and algorithms to solve basic computational problems.
Module Learning Outcomes مخرجات التعلم للمادة الدراسية	Upon Completion this Course, the students will able to: 1. Explain the fundamental concepts of data structures and algorithms. 2. Implement and manipulate one-dimensional and two-dimensional arrays. 3. Explain the concepts and operations of stacks and queues. 4. Explain the structure and operation of linked lists and implement basic linked-list operations. 5. Describe binary trees and binary search trees and implement their basic operations. 6. Represent graphs using appropriate data structures. 7. Implement and analyze basic sorting algorithms, including Bubble Sort and Merge Sort. 8. Select appropriate data structures and algorithms for solving basic computational problems.
Indicative Contents المحتويات الإرشادية	• Introduction to Data Structures and Algorithms, fundamental concepts of data organization, types of data structures, algorithms and their characteristics, and basic problem-solving concepts. (3 hrs) • Arrays, one-dimensional and two-dimensional arrays, array creation and initialization, accessing and updating elements, and basic array manipulation techniques. (3 hrs) • Stack, fundamental concepts of stacks, Last-In-First-Out (LIFO) principle, stack structure, applications of stacks, and basic stack operations. (3 hrs) • Stack Operations, push and pop operations, peek operation, checking stack status, stack traversal, handling overflow and underflow conditions, and practical applications of stack operations. (3 hrs)

	• Stack Implementation, implementation of stacks, array-based implementation, lists for stack representation, and practical programming exercises. (3 hrs) • Queue, fundamental concepts of queues, First-In-First-Out (FIFO) principle, queue structure, applications of queues, and basic queue operations. (3 hrs) • Queue Operations and Implementation, enqueue and dequeue operations, front and rear operations, queue status checking, implementation of queues, and practical queue-based applications. (3 hrs) • Midterm Exam. • Linked List, fundamental concepts of linked lists, nodes and links, structure of singly linked lists, traversal techniques, insertion and deletion concepts, and applications of linked lists. (3 hrs) • Linked List Implementation, implementation of singly linked lists, node creation, insertion and deletion operations, searching and traversal, and practical programming exercises. (3 hrs) • Binary Tree Representation, fundamental concepts and terminology of binary trees, root, nodes, edges, leaves and levels, binary tree representation, and basic tree traversal concepts. (3 hrs) • Binary Search Tree Implementation, concepts of Binary Search Trees (BST), insertion and searching operations, deletion concepts, traversal of binary search trees, and implementation. (3 hrs) • Graph Representation, fundamental concepts and terminology of graphs, vertices and edges, directed and undirected graphs, adjacency matrix and adjacency list representations, and applications of graphs in interconnected systems. (3 hrs) • Bubble Sort, fundamental concepts of sorting, comparison-based sorting, Bubble Sort algorithm, step-by-step implementation, performance considerations, and practical applications. (3 hrs) • Merge Sort, divide-and-conquer strategy, Merge Sort algorithm, splitting and merging procedures, implementation, complexity analysis, and comparison with Bubble Sort. (3 hrs)

Learning and Teaching Strategies استراتيجيات التعلم والتعليم	
Strategies	The module will employ a combination of theoretical and practical teaching strategies to develop students' conceptual understanding and programming skills. The main strategies include: • Interactive Lectures: Presenting fundamental concepts of data structures and algorithms with examples and visual representations.

	• Problem-Based Learning: Introducing programming problems that require students to select and apply appropriate data structures or algorithms. • Practical Programming: Implementing data structures and algorithms using Python through guided laboratory exercises. • Demonstration and Visualization: Using step-by-step examples to illustrate operations such as insertion, deletion, traversal, searching, and sorting. • Hands-on Exercises: Providing individual and group programming tasks to reinforce theoretical concepts. • Discussion and Analysis: Comparing different data structures and algorithms based on their functionality, efficiency, and appropriate use cases. • Formative Assessment: Using quizzes, coding exercises, short assignments, and practical activities to continuously evaluate students' understanding.

Student Workload (SWL)

الحمل الدراسي للطالب محسوب لـ ١٥ اسبوعا

Structured SWL (h/sem) الحمل الدراسي المنتظم للطالب خلال الفصل	**78**	**Structured SWL (h/w)** الحمل الدراسي المنتظم للطالب أسبوعيا	**5.2**
Unstructured SWL (h/sem) الحمل الدراسي غير المنتظم للطالب خلال الفصل	**47**	**Unstructured SWL (h/w)** الحمل الدراسي غير المنتظم للطالب أسبوعيا	**3.1**
Total SWL (h/sem) الحمل الدراسي الكلي للطالب خلال الفصل	**125**		

Module Evaluation

تقييم المادة الدراسية

		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	Quizzes	2	20% (10)	7 and 14	LO #1 and #8
	Assignments	2	10% (5)	11 and 15	LO #2 and #8
	Projects / Lab.	1	10% (10)		
	Report				
Summative assessment	Midterm Exam	1 hr	10% (10)	9	LO #1 - #3
	Final Exam	3 hr	50% (50)	16	All
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus) المنهاج الاسبوعي النظري	
	Material Covered
Week 1	Introduction to Data Structures & Algorithms
Week 2	Arrays (1D & 2D in Python)
Week 3	Stack
Week 4	Stack Operations
Week 5	Stack Implementation
Week 6	Queue
Week 7	Queue Operations & Implementation
Week 8	Midterm Exam
Week 9	Linked List
Week 10	Linked List Implementation
Week 11	Binary Tree Representation
Week 12	Binary Search Tree Implementation
Week 13	Graph Representation
Week 14	Bubble Sort
Week 15	Merge Sort
Week 16	**Preparatory week before the final Exam**

Delivery Plan (Weekly Lab. Syllabus) المنهاج الاسبوعي للمختبر	
	Material Covered
Week 1	Introduction to Data Structures & Algorithms: Introduction to the Python programming environment, basic data structures, algorithm representation, simple problem-solving exercises, and writing basic Python programs.
Week 2	Arrays (1D & 2D in Python): Creating and initializing one-dimensional and two-dimensional arrays, accessing and modifying elements, traversing arrays, and performing basic array operations using Python.
Week 3	Stack: Introduction to stack data structure, implementing a basic stack using Python lists, applying the LIFO principle, and performing simple stack-based programming exercises.
Week 4	Stack Operations: Implementing Push, Pop, Peek, IsEmpty, and Size operations, handling empty-stack conditions, and testing stack operations using Python programs.

Week 5	Stack Implementation: Developing a complete stack implementation in Python, testing stack functionality, handling overflow and underflow conditions, and solving practical problems using stacks.
Week 6	Queue: Introduction to queue data structure, implementing a basic queue using Python lists, applying the FIFO principle, and performing basic queue operations.
Week 7	Queue Operations & Implementation: Implementing Enqueue, Dequeue, Front, Rear, IsEmpty, and Size operations, testing queue functionality, and developing practical queue-based applications in Python.
Week 8	Midterm Exam: Practical assessment covering arrays, stacks, and queues.
Week 9	Linked List: Creating nodes, establishing links between nodes, constructing a singly linked list, traversing the list, and displaying linked-list elements using Python.
Week 10	Linked List Implementation: Implementing insertion, deletion, searching, and traversal operations in a singly linked list using Python.
Week 11	Binary Tree Representation: Creating binary tree nodes, representing binary trees in Python, establishing parent-child relationships, and performing basic tree traversal exercises.
Week 12	Binary Search Tree Implementation: Implementing a Binary Search Tree in Python, performing insertion and search operations, and applying tree traversal techniques.
Week 13	Graph Representation: Creating graphs using adjacency matrices and adjacency lists, adding vertices and edges, and implementing basic graph representation techniques using Python.
Week 14	Bubble Sort: Implementing Bubble Sort in Python, performing sorting operations, testing the algorithm with different datasets, and analyzing its basic performance.
Week 15	Merge Sort: Implementing Merge Sort using the divide-and-conquer approach, performing splitting and merging operations, testing the algorithm, and comparing its performance with Bubble Sort.

Learning and Teaching Resources
مصادر التعلم والتدريس

	Text	Available in the Library?
Required Texts	• Vinu V Das, Principles of Data Structures Using C and C++	No
Recommended Texts	• Ramesh Vasappanavara, Anand Vasappanavara, Data Structures using C by practice. • D.S. MALIK, Data Structures Using C++, second edition, 2nd Edition. • Robert L. Kruse, Alexander J. Ryba, Data Structures and Program Design in C++.	No
Websites		

Grading Scheme مخطط الدرجات				
Group	**Grade**	التقدير	**Marks %**	**Definition**
Success Group (50 - 100)	**A** - Excellent	امتياز	90 - 100	Outstanding Performance
	B - Very Good	جيد جدا	80 - 89	Above average with some errors
	C - Good	جيد	70 - 79	Sound work with notable errors
	D - Satisfactory	متوسط	60 - 69	Fair but with major shortcomings
	E - Sufficient	مقبول	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX** – Fail	راسب (قيد المعالجة)	(45-49)	More work required but credit awarded
	F – Fail	راسب	(0-44)	Considerable amount of work required

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.